IEICE TRANS. INF. & SYST., VOL.E97–D, NO.5 MAY 2014

LETTER

Multiple Kernel Learning for Quadratically Constrained MAP Classification

Yoshikazu WASHIZAWA[†,††a)], ***Member***, **Tatsuya YOKOTA**[††,†††], ***Student Member***, ***and*** **Yukihiko YAMASHITA**[†††], ***Member***

SUMMARY Most of the recent classification methods require tuning of the hyper-parameters, such as the kernel function parameter and the regularization parameter. Cross-validation or the leave-one-out method is often used for the tuning, however their computational costs are much higher than that of obtaining a classifier. Quadratically constrained maximum a posteriori (QCMAP) classifiers, which are based on the Bayes classification rule, do not have the regularization parameter, and exhibit higher classification accuracy than support vector machine (SVM). In this paper, we propose a multiple kernel learning (MKL) for QCMAP to tune the kernel parameter automatically and improve the classification performance. By introducing MKL, QCMAP has no parameter to be tuned. Experiments show that the proposed classifier has comparable or higher classification performance than conventional MKL classifiers.

key words: *quadratically constrained MAP, multiple kernel learning, support vector machine, Bayes classification rule*

1. Introduction

Multiple kernel learning (MKL) methods for classification problems have been researched to tune the kernel parameter automatically as well as to obtain an optimal kernel function from a set of kernel functions [1]. Most of them use a linear combination of positive definite kernels $k_1, \ldots, k_L$, and the discriminant function is given by

$$f(\boldsymbol{x}) = \sum_{i=1}^{M} \alpha_i k_\eta(\boldsymbol{z}_i, \boldsymbol{x}) + \beta = \sum_{i=1}^{M} \alpha_i \sum_{l=1}^{L} \xi_l k_l(\boldsymbol{z}_i, \boldsymbol{x}) + \beta, \tag{1}$$

where $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M$ are basis samples (so-called support vectors in SVM), α_i and β are the weight parameters, $k_\eta(\boldsymbol{x}, \boldsymbol{x}') = \sum_{l=1}^{L} \xi_l k_l(\boldsymbol{x}, \boldsymbol{x}')$ is the kernel function, and $\xi_l \geq 0$ is a kernel weight for the lth kernel. Learning α_i and β has been researched as kernel machines, and MKL learns ξ_l simultaneously. The kernel alignment function $A(\boldsymbol{K}, \boldsymbol{y}\boldsymbol{y}^\top) = \langle \boldsymbol{K}, \boldsymbol{y}\boldsymbol{y}^\top \rangle_F / (M \|\boldsymbol{K}\|_F)$ and/or the structural risk minimization (i.e., maximum margin criterion with regularization) is used as the criteria of MKL [1], where $\boldsymbol{y} = \{-1, +1\}^M$ is the training label vector, K is the kernel Gram matrix, and $\langle \cdot, \cdot \rangle_F$ and $\| \cdot \|_F$ are the Schmidt inner product and the Frobenius norm respectively. By introducing kernel weight learning, MKL obtains optimal ξ_l automatically, however, we still have to tune the regularization parameter.

Quadratically constrained MAP (QCMAP) classification, which has no regularization parameter, was developed [2]. QCMAP is based on the Bayes classification rule and has reported higher classification accuracies than support vector machine (SVM). In QCMAP classifier, we only have to tune the kernel function for each classification problem. We explain QCMAP in Sect. 2 in detail.

In this paper, we therefore propose MKL for QCMAP (MKQCMAP) classifier to realize a hyper-parameter free classifier and improve classification performance. Unlike SVM, the optimization criterion of QCMAP is not defined in the feature space induced by the kernel function, but in the weight parameter space. Thus, the same optimization criterion can be used for both the weight parameter (α_i and β) and the kernel weight parameter ξ_l. We introduce two MKL methods for QCMAP in accordance with the optimization criterion.

If we directly apply the MKL methods to QCMAP, kernel functions that reduce the empirical error tend to be chosen, and this results in the over-fitting problem since QCMAP uses the empirical error minimization with no regularization. For example, if we use Gaussian kernel functions $k_l(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma_l \|\boldsymbol{x} - \boldsymbol{x}'\|^2)$, only the largest γ_l (corresponds to the smallest variance) is chosen. In order to avoid this problem, we introduce the cross-learning method.

We compared two proposals, MKGQCM1 and MKGQCM2, with six conventional classifiers in Sect. 4. The proposed methods exhibit comparable or higher classification performance than the conventional methods.

2. Quadratically constrained MAP classification

Let $y \in \{-1, +1\}$ be the category for binary classification from a pattern $\boldsymbol{x} \in \mathbb{R}^d$. In the Bayesian classification rule, $\boldsymbol{x}$ is classified in accordance with the maximum a posteriori (MAP) criterion,

$$\hat{y} = \underset{y \in \{-1,+1\}}{\operatorname{argmax}} P(y|\boldsymbol{x}) = \underset{y \in \{-1,+1\}}{\operatorname{argmax}} P(y) p(\boldsymbol{x}|y), \tag{2}$$

where $\hat{y}$ is the estimated label. QCMAP introduces the equivalent optimization problem for a discriminant function $W(\boldsymbol{x})$,

Manuscript received September 19, 2013.
Manuscript revised January 14, 2014.

[†]The author is with The Univeristy of Electro-Communications, Chofu-shi, 182–8585 Japan.
[††]The authors are also with Brain Science Institute, RIKEN, Wako-shi, 351–0198 Japan.
[†††]The authors are with Tokyo Institute of Technology, Tokyo, 152–8552 Japan.
a) E-mail: washizawa@uec.ac.jp

DOI: 10.1587/transinf.E97.D.1340

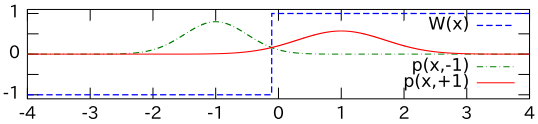

Fig. 1 Example of QCMAP.

$$\begin{aligned} \max \quad & \sum_{y\in\{-1,+1\}} \int_{\mathcal{F}} P(\boldsymbol{x}, y) \min(yW(\boldsymbol{x}), 1) \mathrm{d}\boldsymbol{x} \\ \text{subject to} \quad & \int_{F} Q(\boldsymbol{x}) |W(\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x} \le 1, \end{aligned} \tag{3}$$

where $\mathcal{F}$ is the data domain, $Q(\boldsymbol{x})$ is an arbitrary weight function that satisfies, $\int_{\mathcal{F}} Q(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = 1$ and $Q(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in \mathcal{F}$. Even if $|W(\boldsymbol{x})|$ is grater than 1 at fixed $\boldsymbol{x}$, the objective function never increases compared to the case $|W(\boldsymbol{x})| = 1$. Therefore, for arbitrary $Q(\boldsymbol{x})$, one of the the solutions is

$$W(\boldsymbol{x}) = \begin{cases} 1, & \text{if } P(+1|\boldsymbol{x}) > P(-1|\boldsymbol{x}) \\ -1, & \text{if } P(+1|\boldsymbol{x}) < P(-1|\boldsymbol{x}), \end{cases} \tag{4}$$

that is $W(\boldsymbol{x}) = \hat{y}$. An example for one dimensional case is shown in Fig. 1. The decision boundary is the intersection of two PDFs.

The model of the decision function $W(\boldsymbol{x})$ is given by,

$$W(\boldsymbol{x}) = \sum_{i=1}^{M} \alpha_i k(\boldsymbol{z}_i, \boldsymbol{x}) + \beta = \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle, \tag{5}$$

where $\{\boldsymbol{z}_i\}_{i=1}^{M}$ is the basis vectors, $k(\cdot,\cdot)$ is the kernel function, $\boldsymbol{w} = [\alpha_1, \ldots, \alpha_M, \beta]^\top$, $\boldsymbol{\phi}(\boldsymbol{x}) = [k(\boldsymbol{z}_1, \boldsymbol{x}), \ldots, k(\boldsymbol{z}_M, \boldsymbol{x}), 1]^\top$. Usually all training vectors are used for the basis vectors. However we discriminate them. This is equivalent to the case $L = 1$ in Eq. (1). We replace the expectation for $\boldsymbol{x}$ and y in the objective function (3) to the empirical average for the training samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$. The optimization problem is given as,

$$\begin{aligned} \max \quad & \sum_{i=1}^{N} \min(y_n \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i) \rangle, 1) \\ \text{subject to} \quad & \boldsymbol{w}^\top \boldsymbol{H} \boldsymbol{w} \le 1, \end{aligned} \tag{6}$$

where $\boldsymbol{H}$ is an $(N+1) \times (N+1)$ matrix whose i, j element is $\int_{\mathcal{F}} Q(\boldsymbol{x})\phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$. $\phi_i(\boldsymbol{x})$ denotes the ith element of $\boldsymbol{\phi}(\boldsymbol{x})$. Since $\min(x, 1) = -\max(1 - x, 0) + 1$, the objective function equivalent to minimization of the sum of the hinge loss,

$$f(\boldsymbol{w}) = \sum_{i=1}^{N} h(y_n \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i) \rangle), \quad h(x) = \max(1 - x, 0). \tag{7}$$

Yokota and Yamashita used the relaxed hinge loss function $h_t(x) = \frac{1}{t}\log(1 + \exp(t(1 - x)))$, with $t = 2$ and the primal-dual interior point (PDIP) method to solve the problem (6) [3]. The optimization algorithm is shown in Algorithm 1. As for the weight function $Q(\boldsymbol{x})$, three weight functions are introduced in [2]. In this paper, we focus on the Gaussian weight function because the other weight functions have hyper-parameters. The mean vector and covariance matrix of the Gaussian function are set to be the mean and the covariance for all training samples, respectively.

Algorithm 1 PDIP algorithm for optimization problem (6)

Require: training samples $\{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$, kernel function k, and weight function $Q(\boldsymbol{x})$

Ensure: optimal weight vector $\boldsymbol{w}$ for the problem (6)

1: Obtain $\boldsymbol{H}$, $(\boldsymbol{H})_{i,j} = \int_{\mathcal{F}} Q(\boldsymbol{x})\phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$, Set initial values $\boldsymbol{w} = \boldsymbol{1}$, $z = 1$.

2: Let Lagrangian; $L = f(\boldsymbol{w}) - z(1 - \boldsymbol{w}^\top \boldsymbol{H}\boldsymbol{w})$ and its gradient $\nabla_{\boldsymbol{w},z} L$.

3: **repeat**

4: Obtain updating directions:

$$\begin{aligned} \Delta\boldsymbol{w} &= (\nabla_{\boldsymbol{w}} f(\boldsymbol{w}) + 2z\boldsymbol{H} + \frac{4z}{c}\boldsymbol{H}\boldsymbol{w}\boldsymbol{w}^\top\boldsymbol{H})^{-1}(\nabla_{\boldsymbol{w}} f(\boldsymbol{w}) + 2az\boldsymbol{H}\boldsymbol{w}) \\ \Delta z &= z - az + z/c 2\boldsymbol{w}^\top \boldsymbol{H}\Delta\boldsymbol{w}, \end{aligned}$$

where $a = \min(\sqrt{\|\nabla_{\boldsymbol{w},z} L\|/(M+1)}, 0.5)$, $c = 1 - \boldsymbol{w}^\top\boldsymbol{H}\boldsymbol{w}$

5: Determine updating factor α, $(\boldsymbol{w}, z) \leftarrow (\boldsymbol{w}, z) + \alpha(\Delta\boldsymbol{w}, \Delta z)$, by using the back tracking search or line search.

6: **until** $\|\nabla_{\boldsymbol{w},z} L\|$ converges

The optimization problem (6) is naturally introduced from the Bayesian classification rule. In [2], the problem (3) is used to obtain optimal $\boldsymbol{w}$. In this paper, we use the same optimization problem to solve both $\boldsymbol{w}$ and ξ_l.

3. Multiple Kernel Learning for QCMAP

We propose two MKL methods for QCMAP, they are named MKGQCM1 and MKGQCM2 (Multiple kernel Gaussian quadratically constrained MAP).

3.1 Extended Basis

MKGQCM1 simply extends the basis function,

$$\begin{aligned} W(\boldsymbol{x}) &= \sum_{l=1}^{L}\sum_{i=1}^{M} \alpha_{i,l} k_l(\boldsymbol{z}_i, \boldsymbol{x}) + \beta = \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle \\ \boldsymbol{w} &= [\alpha_{1,1}, \ldots, \alpha_{M,1}, \alpha_{1,2}, \ldots, \alpha_{M,2}, \ldots, \alpha_{M,L}, \beta]^\top \\ \boldsymbol{\phi}(\boldsymbol{x}) &= [k_1(\boldsymbol{z}_1, \boldsymbol{x}), \ldots, k_1(\boldsymbol{z}_M, \boldsymbol{x}), \\ &\qquad k_2(\boldsymbol{z}_1, \boldsymbol{x}), \ldots, k_2(\boldsymbol{z}_M, \boldsymbol{x}), \ldots, k_L(\boldsymbol{z}_M, \boldsymbol{x}), 1]^\top. \end{aligned} \tag{8}$$

In this case, the decision function does not have the form Eq. (1). When we use the boosting for several kernel machines such as LPboost [4], the decision function has the same form as Eq. (8). When we use the Gaussian weight function $Q(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and Gaussian kernel functions, $\boldsymbol{H}$ is given by

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_{1,1} & \ldots & \boldsymbol{H}_{1,L} & \boldsymbol{u}_1 \\ \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{H}_{L,1} & \ldots & \boldsymbol{H}_{L,L} & \boldsymbol{u}_L \\ \boldsymbol{u}_1^\top & \ldots & \boldsymbol{u}_L^\top & 1 \end{bmatrix} \in \mathbb{R}^{(LM+1)\times(LM+1)}, \tag{9}$$

where the (i, j) element of the sub-matrix $\boldsymbol{H}_{l,m} \in \mathbb{R}^{M\times M}$ and the ith element of the vector $\boldsymbol{u}_l \in \mathbb{R}^M$ are respectively given

by

$$(\boldsymbol{H}_{l,m})_{i,j} = \int_{\mathcal{F}} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) k_l(\boldsymbol{z}_i, \boldsymbol{x}) k_m(\boldsymbol{z}_j, \boldsymbol{x}) \mathrm{d}\boldsymbol{x} \tag{10}$$

$$(\boldsymbol{u}_l)_i = \int_{\mathcal{F}} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) k_l(\boldsymbol{z}_i, \boldsymbol{x}) \mathrm{d}\boldsymbol{x}, \tag{11}$$

where $(\cdot)_{i,j}$ and $(\cdot)_i$ are the (i, j) element of a matrix and the ith element of a vector respectively. Since the multiplication of Gaussian functions is also a Gaussian function, these values can be analytically calculated by the Gaussian integral formula.

The problem can be solved by the same algorithm as Algorithm 1. The number of parameters is $LM + 1$ that is much larger than that of the original QCMAP. When L or M is large, its computational complexity is also high.

3.2 Alternating Optimization

MKGQCM2 alternately optimizes $\boldsymbol{w}$ and $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_L]^\top$ in Eq. (1). For fixed $\boldsymbol{\xi}$, the optimization problem for $\alpha_1, \ldots, \alpha_M$ and β has the same form as the problem (6), where $\boldsymbol{\phi}(\boldsymbol{x}) = [k_\eta(\boldsymbol{z}_1, \boldsymbol{x}), \ldots, k_\eta(\boldsymbol{z}_M, \boldsymbol{x})]^\top$. This can be solved by the same PDIP algorithm 1. When we use the Gaussian weight function $Q(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and Gaussian kernel functions, the (i, j)-element of $\boldsymbol{H}$ is given as

$$(\boldsymbol{H})_{i,j} = \begin{cases} \sum_{l=1}^{L} \sum_{m=1}^{L} \xi_l \xi_m (\boldsymbol{H}_{l,m})_{i,j} & (i, j = 1, \ldots, M) \\ \sum_{l=1}^{L} \xi_l (\boldsymbol{u}_l)_j & (i = M+1, j \neq i) \\ \sum_{l=1}^{L} \xi_l (\boldsymbol{u}_l)_i & (j = M+1, i \neq j) \\ 1 & (i = j = M+1) \end{cases}.$$

For fixed $\boldsymbol{w}$, optimization problem for $\boldsymbol{\xi}$ is given as

$$\begin{array}{ll} \min_{\boldsymbol{\xi}} & f(\boldsymbol{\xi}) = \sum_{i=1}^{N} h\Big(y_i w_{M+1} + y_i \langle \boldsymbol{\xi}, \boldsymbol{\psi}_n \rangle\Big) \\ \text{subject to} & f_1(\boldsymbol{\xi}) = \langle \boldsymbol{\xi}, \tilde{\boldsymbol{H}} \boldsymbol{\xi} \rangle + 2\langle \boldsymbol{\xi}, \tilde{\boldsymbol{h}} \rangle + w_{M+1} - 1 \leq 0 \\ & f_{l+1}(\boldsymbol{\xi}) = -\xi_l \leq 0, \quad l = 1, \ldots, L, \end{array} \tag{12}$$

where

$$(\boldsymbol{\psi}_n)_l = \sum_{m=1}^{M} w_m k_l(\boldsymbol{z}_m, \boldsymbol{x}_n), \ (\tilde{\boldsymbol{H}})_{l,m} = \sum_{i=1}^{M} \sum_{j=1}^{M} w_i w_j (\boldsymbol{H}_{l,m})_{i,j}$$

$$(\tilde{\boldsymbol{h}})_l = w_{M+1} \sum_{i=1}^{M} w_i (\boldsymbol{u}_l)_i.$$

The objective function and the first constraint are exactly equivalent to the problem (6). This problem is also convex, and can be solved by PDIP algorithm. The algorithm procedure is shown in Algorithm 2.

This alternating optimization converges to a local minimum of the problem (6). $\boldsymbol{w}$ and $\boldsymbol{\xi}$ are initialized as $w_i = 1$ for $i = 1, \ldots, M + 1$ and $\xi_l = 1/L$ for $l = 1, \ldots, L$. We first optimize $\boldsymbol{w}$ with fixed $\boldsymbol{\xi}$, and then alternately optimize $\boldsymbol{\xi}$ and $\boldsymbol{w}$ until the value of the objective function converges.

Algorithm 2 PDIP algorithm for optimizing $\boldsymbol{\xi}$

Require: training samples $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$, weight function $Q(\boldsymbol{x})$, weight $\boldsymbol{w}$
Ensure: optimal kernel weight $\boldsymbol{\xi}$ for the problem (12)
1: Obtain $\tilde{\boldsymbol{H}}, \tilde{\boldsymbol{h}}, \boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_N$
2: Set initial value for $\xi_l = 1/L$, $l = 1, \ldots, L$, Lagrange multiplier $\lambda_l = 1$, $l = 1, \ldots, L + 1$, parameter μ is commonly from 10 to 20 [3].
3: Let $\boldsymbol{f}(\boldsymbol{\xi}) = [f_1(\boldsymbol{\xi}), \ldots, f_{L+1}(\boldsymbol{\xi})]^\top$, $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_{L+1}]^\top$, $\boldsymbol{D}_{\boldsymbol{f}}(\boldsymbol{\xi}) = [\nabla f_1(\boldsymbol{\xi}) \cdots \nabla f_{L+1}(\boldsymbol{\xi})]^\top = [2(\tilde{\boldsymbol{H}}\boldsymbol{\xi} + \tilde{\boldsymbol{h}})| - \boldsymbol{I}_L]^\top$.
4: **repeat**
5: Obtain $\tau = -\mu(L + 1)/\boldsymbol{\xi}^\top \boldsymbol{\lambda}$
6: Obtain updating directions $(\Delta\boldsymbol{\xi}, \Delta\boldsymbol{\lambda})$ by solving the Newton-Raphson linear equation;

$$\begin{bmatrix} \nabla^2 f(\boldsymbol{\xi}) + \sum_{i=1}^{L+1} \nabla^2 f_i(\boldsymbol{\xi}) & \boldsymbol{D}_{\boldsymbol{f}}(\boldsymbol{\xi})^\top \\ -\mathrm{diag}(\boldsymbol{\xi}) \boldsymbol{D}_{\boldsymbol{f}}(\boldsymbol{\xi}) & -\mathrm{diag}(\boldsymbol{f}(\boldsymbol{\xi})) \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\xi} \\ \Delta\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \nabla f(\boldsymbol{\xi}) + \boldsymbol{D}_{\boldsymbol{f}}(\boldsymbol{\xi})^\top \boldsymbol{\lambda} \\ -\mathrm{diag}(\boldsymbol{\lambda}) \boldsymbol{f}(\boldsymbol{\xi}) - \frac{1}{\tau} \boldsymbol{1}_{L+1} \end{bmatrix}$$

7: Determine updating factor s, $(\boldsymbol{\xi}, \boldsymbol{\lambda}) \leftarrow (\boldsymbol{\xi}, \boldsymbol{\lambda}) + s(\Delta\boldsymbol{\xi}, \Delta\boldsymbol{\lambda})$, by using the back tracking search or line search.
8: **until** $\|\nabla_{\boldsymbol{\xi},\boldsymbol{\lambda}} (f(\boldsymbol{\xi}) + \sum_{i=1}^{L+1} \lambda_i f_i(\boldsymbol{\xi}))\|$ converges.

3.3 Cross Learning Method

If all training samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ are used for the basis, the learning step for $\boldsymbol{\xi}$ tends to select a complex decision boundary because the empirical hinge loss is used for the objective function (7). In the case of Gaussian kernel, parameters corresponding to lager γ (i.e. smaller variance) tend to have larger values. For instance, when $\gamma \to \infty$, elements of $\boldsymbol{H}$ become zero, and $k(\boldsymbol{x}, \boldsymbol{z}) = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{z}\|^2)$ has value one only if $\boldsymbol{x} = \boldsymbol{z}$, otherwise zero. In this case, $\alpha_i = 2y_i, \beta = -1$ is one of the solutions that gives $f(\boldsymbol{w}) = 0$. This causes an over-fitting problem.

In order to avoid this problem, we introduce the cross-learning method that uses different sets for the basis vectors $\{\boldsymbol{z}_i\}_{i=1}^M$ and for the evaluation of the hinge loss. In particular, we split given N training samples to K subsets. Then for each subset, we obtain the classifier parameters $(\boldsymbol{w}_k, \boldsymbol{\xi}_k)$ for $k = 1, \ldots, K$, where vectors in the subset are used for the basis $\{\boldsymbol{z}_i\}_{i=1}^M$, and all training vectors are used to evaluate the hinge loss. Finally, we calculate the average of output values of K classifiers for query vectors.

4. Experiment

We used 13 classification benchmark datasets used in [2]. 'image' and 'splice' datasets have 20 realizations (training and test sets), and the others have 100 realizations. We compared the proposed methods (MKGQCM1 and MKGQCM2) with GQCM [2], SVM, Semi-Infinite Linear Programming (SILP) [5], LpMKL ($p = 2$) [6], LPboost [4], and the linear discriminant analysis (LDA)[†]. LDA is a clas-

[†]We used the following software; SVM, and LPboost: LibSVM [7] (`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`); SILP and LpMKL: SHOGUN (`http://shogun-toolbox.org/`); LPboost: GLPK (`http://www.gnu.org/software/glpk/`) for the linear programming;

Table 1 Mean classification error rate [%] and standard deviation; the lowest error rates are underlined. the second row is the number of hyper-parameters. the last row is the averaged rank.

Data	MKGQCM1	MKGQCM2	GQCM	SVM	SILP	LpMKL	LPBoost	LDA
No. hyper-param.	0	0	1	2	1	1	1	0
banana	11.18± 0.48	10.94± 0.45	10.80±0.66	10.79±0.77	11.31±4.84	10.86±0.57	14.41±1.53	46.61±4.80
breast-cancer	28.82± 5.00	27.75± 4.55	26.70±5.11	26.88±5.08	28.44±4.40	27.13±4.57	29.57±4.62	33.73±4.69
diabetis	25.20± 2.08	23.67± 1.99	23.71±1.84	24.09±2.66	24.52±2.19	23.83±1.69	32.08±2.84	24.98±1.95
flare-solar	36.42± 1.80	35.08± 1.76	33.66±1.94	33.13±2.49	35.70±2.48	35.83±1.76	34.66±2.61	34.65±1.72
german	24.57± 2.42	23.59± 2.13	24.30±2.16	24.30±2.29	26.22±2.48	23.47±2.23	28.04±2.39	28.74±2.56
heart	17.65± 3.27	15.79± 3.57	16.31±3.29	18.88±8.11	17.34±4.08	16.69±3.37	24.32±3.96	16.52±2.99
image	2.96± 0.52	4.89± 0.82	2.97±0.68	3.42±0.81	3.11±0.79	3.06±0.63	2.90±0.70	17.68±0.62
ringnorm	1.55± 0.13	1.71± 0.18	1.77±0.48	1.61±0.19	1.57±0.13	1.57±0.14	3.24±5.81	24.62±0.65
splice	12.76± 0.72	14.95± 0.78	10.90±0.71	11.02±0.65	11.41±0.66	12.58±0.74	12.08±1.39	16.24±0.58
thyroid	4.65± 2.13	4.45± 2.20	4.36±2.27	5.16±2.36	5.24±2.20	4.20±2.31	5.13±4.17	13.37±3.35
titanic	22.52± 1.24	22.67± 0.94	22.33±0.75	22.70±1.57	22.85±1.68	22.69±1.46	22.84±1.79	23.26±1.48
twonorm	2.68± 0.20	2.39± 0.14	2.55±0.30	2.87±1.60	2.82±3.39	2.42±0.15	5.24±7.53	2.61±0.17
waveform	10.11± 0.50	10.54± 0.89	10.54±1.03	10.27±0.65	10.18±0.61	9.74±0.47	12.11±0.96	17.39±0.63
No. underlined	1	3	3	2	0	3	1	0
Avr. rank	4.54	3.69	2.69	4.00	5.08	3.15	6.08	6.77

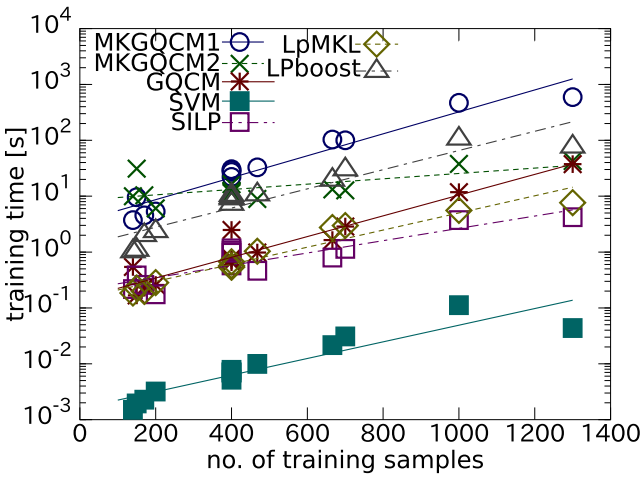

Fig. 2 Training time.

sifier with no hyper-parameters. Gaussian kernel functions with $\log_{10}(\gamma) = -6, -5.75, -5.50, \ldots, 1$ were used. For SVM, SILP, and LpMKL, the regularization parameter was chosen from $\log_{10}(C) = -3, -2.75, \ldots, 3$ by the cross-validation. For LPboost, SVMs using all combination of γ and C were used for the weak learner. For GQCM and SVM, one kernel parameter was chosen by the cross-validation for each realization in the training set. The Gaussian weight function was used for MKGQCM1, MKGQCM2, and GQCM. For the cross-learning method, $K = 10$ was used. Table 1 shows the classification errors and standard deviations. Among 13 datasets, MKGQCM2 exhibited the best classification performance in three datasets that is the same as GQCM and LpMKL. For averaged rank, GQCM was the best score (2.69), and the proposed MKGQCM2 was the third best (3.69).

Figure 2 shows the relation between the number of training samples and training time to obtain a classifier. Averaged training times for each dataset and regression lines are shown. MKGQCM2 was faster than MKGQCM1, and its gradient is smaller than the other methods because of the cross-learning. For instance, although SILP is about ten times faster than MKGQCM2 when $N = 1300$, SILP requires to obtain (10 CVs) $\times$ (25 values for hyper-parameter) classifiers to obtain the optimal hyper-parameter before obtaining the final classifier. Therefore, the total computational cost of SILP is higher than that of MKGQCM2. In SVM, since the number of hyper-parameters is two (for kernel function and regularization), the number of combinations becomes much larger.

5. Conclusion

We have proposed multiple kernel learning for the quadratically constrained MAP classification. We introduced two learning methods, MKGQCM1 and MKGQCM2. MKGQCM1 simply extends the basis function, and MKGQCM2 alternately updates the weight vector $\boldsymbol{w}$ and kernel weight parameter $\boldsymbol{\xi}$. Our experiment showed that MKGQCM2 was faster than MKGQCM1 while offering better classification performance.

By introducing the multiple kernel learning, our classifiers do not require hyper-parameters to be tuned for each dataset. This property is useful and reduces the computational complexity due to the hyper-parameter selection, such as the cross-validation and the leave-one-out. Furthermore, MKGQCM2 exhibited comparable or better performance than SVM and recent MKL methods.

References

[1] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," J. Machine Learning Research, vol.12, pp.2211–2268, 2011.
[2] T. Yokota and Y. Yamashita, "A quadratically constrained MAP classifier using the mixture of Gaussian models as a weighted function," IEEE Trans. Neural Networks and Learning Systems, vol.24, no.7, pp.1127–1140, 2013.
[3] S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge Univeristy Press, 2004.
[4] A. Demiriz, K. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation," J. Machine Learning Research, vol.46, no.1-3, pp.225–254, 2002.

[5] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Learge scale multiple kernel learning," J. Machine Learning Research, vol.7, pp.1531–1565, 2006.
[6] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien, "l_p-norm multiple kernel learning," J. Machine Learning Research, vol.12, pp.953–997, 2011.
[7] C.C. Chang and C.J. Lin, "LIBSVM: A library for support vector machines," ACM Trans. Intelligent Systems and Technology, vol.2, no.27, pp.1–27, 2011.